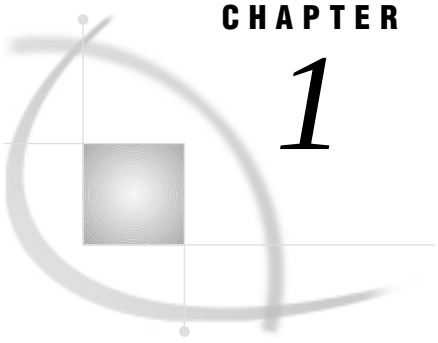

CHAPTER

1

DB2 under UNIX and PC Hosts Chapter, First Edition

Introduction **1**
SAS/ACCESS LIBNAME STATEMENT **2**
Data Set Options: DB2 Specifics **10**
ACCESS Procedure: DB2 Specifics **19**
ACCESS Procedure Statements for DB2 **19**
DBLOAD Procedure: DB2 Specifics **20**
DBLOAD Procedure Statements for DB2 **21**
DBLOAD Procedure Examples **22**
SQL Procedure Pass-Through Facility: DB2 Specifics **23**
DB2 Naming Conventions **25**
DB2 Data Types **26**
String Data **26**
Numeric Data **27**
Dates, Times, and Timestamps **27**
Null Values **28**
ACCESS Procedure Data Conversions **28**
DBLOAD Procedure Data Conversions **29**
LIBNAME Statement Data Conversions **29**

Introduction

This chapter introduces SAS System users to Database 2 (DB2), IBM's relational database management system. * DB2 runs under the UNIX, Microsoft Windows, and OS/2 operating environments. This chapter accompanies and should be used with *SAS/ACCESS Software for Relational Databases: Reference* (order #57204).**

This chapter describes the SAS/ACCESS LIBNAME and data set options that are specific to DB2. It then focuses on the terms and concepts that will help you use the SAS/ACCESS Interface to DB2. Finally, it describes the statements that are specific to DB2 that you use in the ACCESS and DBLOAD procedures and in the SQL procedure's CONNECT statement. All platforms may not have all three of these procedures. See the procedure descriptions for further information.

The SAS/ACCESS Interface to DB2 uses the Call Level Interface, or CLI. For more information on customizing your SAS application, refer to your vendor-specific documentation on the CLI interface.

For general information on database management systems, including information for the database administrator on how the SAS/ACCESS interface works, see Appendix 2, "DBMS Overview and Information for the Database Administrator" .

* IBM previously called this product OS/2 Database Manager as well as DB2 for Common Servers.

SAS/ACCESS LIBNAME STATEMENT

This section describes the LIBNAME statement and its options that are specific to DB2. The LIBNAME statement and options that are common to most databases are fully described in Chapter 3, "SAS/ACCESS LIBNAME Statement".

LIBNAME Statement: DB2 Specifics

Associates a SAS libref with a DBMS database, schema, server, or group of tables and views.

Valid: in a DATA or PROC step

Syntax

LIBNAME *libref SAS/ACCESS-engine-name*
SAS/ACCESS-engine-connection-options
<*SAS/ACCESS-LIBNAME-options*>;

Arguments

libref
is any SAS name that serves as an alias to associate the SAS System with a database.

SAS/ACCESS-engine-name
is a SAS/ACCESS engine name for your DBMS, in this case, DB2. SAS/ACCESS engines are implemented differently in different operating environments. The engine name is required.

SAS/ACCESS-engine-connection-options
are options that you specify in order to connect to a particular database; these options are different for each database. If the SAS/ACCESS engine connection options contain characters that are not allowed in SAS names, enclose the values of the options in quotation marks. If you specify the appropriate system options or environment variables for your database prior to invoking SAS, you can often omit the SAS/ACCESS engine connection options.

SAS/ACCESS-LIBNAME-options
are options that apply to the objects in a DBMS, such as its tables or indexes. For example, the ROWSET_SIZE= option enables you to specify the number of rows to use when reading data from the DBMS. Support for many of these options is DBMS specific.

Some SAS/ACCESS LIBNAME options can also be specified as SAS/ACCESS engine data set options. When you specify an option in the LIBNAME statement, it applies to objects in the particular database (which is accessed by the libref). A SAS/ACCESS data set option applies only to the data set on which it is specified. If a like named option is specified in both the SAS/ACCESS engine LIBNAME statement and after a data set name (which represents a DBMS table or view), the SAS System uses the value that is specified after the data set name.

For more information, see Chapter 3, "SAS/ACCESS LIBNAME Statement".

Details The LIBNAME statement associates a libref with a SAS/ACCESS engine in order to access tables or views in a database management system (DBMS). The SAS/ACCESS engine enables you to connect to a particular DBMS and, therefore, to specify a DBMS table or view name in a two-level SAS name. For example, in MYLIB.EMPLOYEES_Q2, MYLIB is a SAS libref that points to a particular DBMS, and EMPLOYEES_Q2 is a DBMS table name. When you specify MYLIB.EMPLOYEES_Q2 in a DATA step or procedure, you dynamically access the DBMS table. Beginning in Version 7, SAS software supports reading, updating, creating, and deleting DBMS tables when the LIBNAME engine is used.

See for more information on options that you can use in the LIBNAME statement.

SAS/ACCESS Engine Connection Options

This section describes the connection options for DB2. The connection options are as follows:

USER= on page 3
PASSWORD= on page 3
DATASRC= on page 3
COMPLETE= on page 4
NOPROMPT= on page 4
PROMPT= on page 4
REQUIRED= on page 4
AUTOCOMMIT= on page 4

There are multiple ways that you can connect to the DBMS when using the LIBNAME statement. Use only one of the following methods for each connection since they are mutually exclusive:

- specify USER=, PASSWORD=, and DATABASE=, or
- specify COMPLETE=, or
- specify NOPROMPT=, or
- specify PROMPT=, or
- specify REQUIRED=.

USER=<'>*username*<'>
enables you to connect to a DB2 database with a user ID that is different from the default ID.

The USER= and PASSWORD= connections are optional in DB2. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID for your operating environment is used.

USER= can also be specified with the UID= alias.

PASSWORD=<'>*password*<'>
specifies the DB2 password that is associated with your DBMS user ID.

The USER= and PASSWORD= connection options are optional in DB2 because users may have default user IDs. If you specify USER=, you must specify PASSWORD=.

PASSWORD= can also be specified with the PW=, USING=, PASS=, and PWD= aliases.

DATASRC=<'>*data-source-name*<'>
specifies the DB2 data source or database to which you want to connect.

DATASRC= is an optional connection option. If you omit it, you connect by using a default environment variable.

DATASRC= can also be specified with the DSN=, DS=, and DATABASE= aliases.

AUTOCOMMIT=YES | NO
indicates whether or not updates are committed immediately after they are submitted.

If AUTOCOMMIT=NO, the SAS/ACCESS engine does the commit automatically when it reaches the end of the file. This is the default for everything except for the SQL Procedure Pass-Through Facility and read-only connections.

If AUTOCOMMIT=YES, no rollback is possible. This is the default for the SQL Procedure Pass-Through Facility and read-only connections.

COMPLETE=<'>*CLI-connection-string*<'>
specifies connection options for your data source or database. If you specify enough correct connection options, the SAS/ACCESS engine connects to your data source or database. Otherwise, you are prompted for the connection options with a dialog box that displays the values from the COMPLETE= connection string. You can edit any field before you connect to the data source. You separate multiple options with a semicolon. When a successful connection is made, the complete connect string is returned in the SYSDBMSG macro variable.

COMPLETE= is similar to the PROMPT= option. However, if COMPLETE= attempts to connect and fails, then a dialog box is displayed and you can edit values or enter additional values. COMPLETE= is optional.

See your driver documentation for more details.

NOPROMPT=<'>*CLI-connection-string*<'>
specifies connection options for your data source or database. You separate multiple options with a semicolon. If you specify enough correct connection options, the SAS/ACCESS engine connects to the data source or database. Otherwise, an error is returned and no dialog box is displayed. NOPROMPT= is optional. If connection options are not specified, the default settings are used.

PROMPT=<'> *CLI-connection-string*<'>
specifies connection options to the data source.

A dialog box is displayed, using the values from the PROMPT= connection string. You can edit any field before you connect to the data source. When a successful connection is made, the complete connect string is returned in the SYSDBMSG macro variable.

PROMPT= is similar to the COMPLETE= option. However, unlike COMPLETE=, PROMPT= does not attempt to connect to the DBMS first. It displays the dialog box where you can edit or enter additional values. PROMPT= is optional.

REQUIRED=<'>*CLI-connection-string* <'>
specifies connection options for your data source or database. You separate multiple options with a semicolon.

If you specify enough correct connection options, such as user ID, password, and data source name, the SAS/ACCESS engine connects to the data source or database. Otherwise, a dialog box is displayed to prompt you for the connection options. Options in the dialog box that are not related to the connection are disabled. REQUIRED= only allows you to modify required fields in the dialog box. When a successful connection is made, the complete connect string is returned in the SYSDBMSG macro variable.

REQUIRED= is similar to COMPLETE= because it attempts to connect to the DBMS first. However, if REQUIRED= attempts to connect and fails, then a dialog box is displayed and you can only edit values that are in the required fields. REQUIRED= is optional.

SAS/ACCESS LIBNAME Options The SAS/ACCESS interface to DB2 supports all of the SAS/ACCESS LIBNAME options listed in Chapter 3, "SAS/ACCESS LIBNAME

Statement". In addition to the supported options, the following LIBNAME options are used only in the interface to DB2 or have DB2–specific aspects to them:

CURSOR_TYPE on page 5

DBINDEX= on page 5

PRESERVE_COL_NAMES= on page 5

PRESERVE_TAB_NAMES= on page 6

QUERY_TIMEOUT= on page 6

READ_ISOLATION_LEVEL= on page 6

READ_LOCK_TYPE= on page 7

ROWSET_SIZE= on page 7

SCHEMA= on page 8

SPOOL= on page 8

STRINGDATES= on page 8

TRACE= on page 8

TRACEFILE= on page 8

UPDATE_ISOLATION_LEVEL= on page 8

UPDATE_LOCK_TYPE = on page 10

CURSOR_TYPE=DYNAMIC | FORWARD_ONLY | KEYSET_DRIVEN | STATIC
specifies the cursor type for read-only and updatable cursors. Not all drivers support all cursor types. An error is returned if the specified cursor type is not supported.

By default, CURSOR_TYPE=DYNAMIC, but the driver is allowed to modify the default without an error.

If CURSOR_TYPE=DYNAMIC, then the cursor reflects all of the changes that are made to the rows in a result set as you scroll around the cursor. The data values and the membership of rows in the cursor can change dynamically on each fetch.

If CURSOR_TYPE=FORWARD_ONLY, then the cursor behaves like a DYNAMIC cursor except that it only supports fetching the rows sequentially.

If CURSOR_TYPE=KEYSET_DRIVEN, then the cursor determines which rows belong to the result set when the cursor is opened. However, changes that are made to these rows will be reflected as you scroll around the cursor.

If CURSOR_TYPE=STATIC, then the cursor builds the complete result set when the cursor is opened. No changes that are made to the rows in the result set after the cursor is opened will be reflected in the cursor. Static cursors are read-only.

CURSOR_TYPE= can also be specified with the CURSOR= alias.

DBINDEX=YES | NO
indicates whether or not SAS calls DB2 to find all indexes that are on the specified table.

Default value: YES.

For a full description of this option, refer to Chapter 3, "SAS/ACCESS LIBNAME Statement".

PRESERVE_COL_NAMES=YES|NO
preserves spaces, special characters, and mixed case in DBMS column names.

Default value: NO

The default value for PRESERVE_COL_NAMES= under DB2 is NO because DB2 is case insensitive and all names default to uppercase. For a full description of this option, refer to Chapter 3, "SAS/ACCESS LIBNAME Statement".

PRESERVE_TAB_NAMES=YES|NO
preserves spaces, special characters, and mixed case in DBMS table names.

Default value: NO

The default value for PRESERVE_TAB_NAMES= under DB2 is NO because DB2 is case insensitive and all names default to uppercase. For a full description of this option, refer to Chapter 3, "SAS/ACCESS LIBNAME Statement".

QUERY_TIMEOUT=*number-of-records*
specifies the number of seconds of inactivity to wait before canceling a query.

Default value: 0

The default value of 0 indicates that there is no time limit for a query. This option is useful when you are testing a query or if you suspect that a query might contain an endless loop.

QUERY_TIMEOUT= can also be specified with the TIMEOUT= alias.

READ_ISOLATION_LEVEL= RR | RS | CS | UR
defines the degree of isolation of the current application process from other concurrently running application processes. The isolation levels are as follows and are thoroughly described below:

RR = Repeatable Read

RS = Read Stability

CS = Cursor Stability

UR = Uncommitted Read

Default value: CS
The degree of isolation identifies

- the degree with which rows that are read and updated by the current application are available to other concurrently executing applications
- the degree with which update activity of other concurrently executing application processes can affect the current application.

The DB2 database manager supports four isolation levels. Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. Thus, all isolation levels ensure that any row that is changed by this application process during a unit of work is not changed by any other application process until the unit of work is complete. The isolation levels are defined in terms of several possible occurrences:

- Dirty read — A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it will be able to see changes made that are by those concurrent transactions even before they commit.

  For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

- Nonrepeatable read — If a transaction exhibits this phenomenon, it is possible that it may read a row once and, if it attempts to read that row again later in the course of the same transaction, the row might have been changed or even deleted by another concurrent transaction. Therefore, the read is not (necessarily) repeatable.

  For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again.

Transaction T1 has now retrieved the same row twice but has seen two different values for it.

- Phantom reads — When a transaction exhibits this phenomemon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

  For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it will see a row that did not previously exist, a phantom.

The isolation levels for READ_ISOLATION_LEVEL= include the following:

- Repeatable Read (RR)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - does not allow phantom reads
- Read Stability (RS)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - allows phantom reads
- Cursor Stability (CS)
  - does not allow dirty reads
  - allows nonrepeatable reads
  - allows phantom reads

  This is the default value for DB2.
- Uncommitted Read (UR)
  - allows dirty reads
  - allows nonrepeatable reads
  - allows phantom reads

READ_ISOLATION_LEVEL= is ignored if READ_LOCK_TYPE= is not set to ROW.

READ_ISOLATION_LEVEL= can also be specified with the RIL= alias.

See Also: UPDATE_ISOLATION_LEVEL= on page 8.

READ_LOCK_TYPE=ROW | TABLE

specifies how DB2 tables are locked during a READ operation.

Default value: ROW

If READ_LOCK_TYPE=ROW, the row is locked for read operations. This prevents concurrent reads on a row.

If READ_LOCK_TYPE=TABLE, the table is locked for read operations. This prevents concurrent reads on a table.

For a full description of this option, refer to Chapter 3, "SAS/ACCESS LIBNAME Statement".

See also: UPDATE_LOCK_TYPE= on page 10.

ROWSET_SIZE=*number-of-rows*

specifies the number of rows to use when reading data from the DBMS.

Default value: 0

When ROWSET_SIZE=0, no internal SAS buffering is performed. Setting ROWSET_SIZE=0 causes the SQLFetch API call to be used.

When ROWSET_SIZE=1, only one row is retrieved at a time. The higher the value for ROWSET_SIZE=, the more rows the DB2 engine retrieves in one fetch operation. This option reduces the amount of I/O that is used and can help

improve performance. However, because SAS software stores the rows in memory, higher values for ROWSET_SIZE= use more memory. In addition, if too many rows are selected at once, then the rows that are returned to the SAS application might be out of date. For example, if someone else modified the rows, you would not see the changes. Setting ROWSET_SIZE=1 or greater causes the SQLExtendedFetch API call to be used.

SCHEMA=*schema-name*
enables you to read database objects, such as tables and views, in the specified schema.

SCHEMA= is optional. If it is omitted, you connect to the default schema, which is your user ID. In the following LIBNAME statement example, the SCHEMA= option causes any reference in SAS to **mydb.employee** to be interpreted by DB2 as **scott.employee**.

```
libname mydb db2 SCHEMA=scott;
```

SCHEMA= may also be specified with the OWNER= alias.

SPOOL=YES | NO
specifies whether or not SAS creates a utility spool file during read operations that are performed with the specified LIBNAME.

Default value: YES

For a full description of this option, refer to Chapter 3, "SAS/ACCESS LIBNAME Statement".

STRINGDATES=YES | NO
specifies whether or not to read date and time values from the DB2 database as character strings or as numeric date values.

Default value: NO

If STRINGDATES=YES, then the SAS application reads date-time values as character strings, 'YYYY-MM-DD'.

If STRINGDATES=NO, then the SAS application reads date-time values as numeric date values.

STRINGDATES=NO is used for Version 6 compatibility.

STRINGDATES= can also be specified with the STRDATES= alias.

TRACE=YES | NO
specifies whether or not to turn on tracing information that is used in debugging.

Default value: NO

If TRACE=YES, tracing is turned on, and the DB2 driver manager writes each function call to the trace file that is specified by TRACEFILE=.

If TRACE=NO, tracing is not turned on.

See also: TRACEFILE= on page 8

TRACEFILE=*filename*
specifies the filename to which the DB2 driver manager writes trace information.

Default value: none

TRACEFILE= is used only when TRACE=YES.

See also: TRACE= on page 8.

UPDATE_ISOLATION_LEVEL= CS | RS | RR
defines the degree of isolation of the current application process from other concurrently running application processes. The isolation levels are as follows and are thoroughly described here:

CS = Cursor Stability

RS = Read Stability

RR = Repeatable Read

Default value: CS
The degree of isolation identifies

- the degree with which rows that are read and updated by the current application are available to other concurrently executing applications
- the degree with which update activity of other concurrently executing application processes can affect the current application.

The DB2 database manager supports three isolation levels. Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. Thus, all isolation levels ensure that any row that is changed by this application process during a unit of work is not changed by any other application process until the unit of work is complete. The isolation levels are defined in terms of several possible occurrences:

- Dirty read — A transaction that exhibits this phenomenon has a very minimal isolation from concurrent transactions. In fact, it will be able to see changes that are made by those concurrent transactions even before they commit.

  For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.
- Nonrepeatable read — If a transaction exhibits this phenomenon, it is possible that it may read a row once and, if it attempts to read that row again later in the course of the same transaction, the row might have been changed or even deleted by another concurrent transaction. Therefore, the read is not (necessarily) repeatable.

  For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.
- Phantom reads — When a transaction exhibits this phenomemon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

  For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it will see a row that did not previously exist, a phantom.

The isolation levels for UPDATE_ISOLATION_LEVEL= include the following:

- Repeatable Read (RR)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - does not allow phantom reads
- Read Stability (RS)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - allows phantom reads
- Cursor Stability (CS)
  - does not allow dirty reads
  - allows nonrepeatable reads
  - allows phantom reads

This is the default value for DB2.

UPDATE_ISOLATION_LEVEL= is ignored if UPDATE_LOCK_TYPE= is not set to ROW.

UPDATE_ISOLATION_LEVEL= may also be specified with the UIL= alias.

See Also: READ_ISOLATION_LEVEL= on page 6.

UPDATE_LOCK_TYPE = ROW|TABLE

specifies how a DB2 table is locked during an UPDATE operation.

Default value: ROW

If UPDATE_LOCK_TYPE=ROW, the row is locked for update operations. This prevents concurrent updates on a row.

If UPDATE_LOCK_TYPE=TABLE, the table is locked for update operations. This prevents concurrent updates on a table.

For a full description of this option, refer to Chapter 3, "SAS/ACCESS LIBNAME Statement".

See also: READ_LOCK_TYPE= on page 7.

Example: Specifying a LIBNAME Statement to Access DB2 Data

In this example, the libref MYDBLIB uses the DB2 engine to connect to a DB2 database by using the SAS/ACCESS engine connection option NOPROMPT=. PROC PRINT is used to display the contents of the DB2 table CUSTOMERS.

```
libname mydblib db2
   noprompt="user=testuser;password=testpass;database=testdb;"

proc print data=mydblib.customers;
   where state='CA';
run;
```

Data Set Options: DB2 Specifics

This section describes options that can be applied to SAS data sets that access data in DB2 tables and views. In some cases, the option is fully described in Chapter 4, "SAS/ACCESS Data Set Options", except for some detail that is specific to DB2, such as a default value. In other cases, the entire option is specific to DB2, so it is fully described in this chapter.

When specified in a DATA step or SAS procedure, the following data set options can be used on a SAS data set that accesses data in a DBMS object, such as a table or view. A data set option applies only to the SAS data set on which it is specified.

The SAS/ACCESS interface to DB2 supports all of the SAS/ACCESS data set options listed in Chapter 4, "SAS/ACCESS Data Set Options" . In addition to the supported options, the following data set options are used only in the interface to DB2 or have DB2–specific aspects to them:

"CURSOR_TYPE=" on page 11

"DBINDEX=" on page 11

"DBNULL=" on page 12

"DBSASTYPE=" on page 12

"DBTYPE=" on page 13

"QUERY_TIMEOUT=" on page 13

"READ_ISOLATION_LEVEL=" on page 14

"READ_LOCK_TYPE=" on page 15
"ROWSET_SIZE=" on page 16
"SASDATEFMT=" on page 16
"SCHEMA=" on page 16
"UPDATE_ISOLATION_LEVEL=" on page 17
"UPDATE_LOCK_TYPE=" on page 18

CURSOR_TYPE=

Specifies the cursor type for read only and updatable cursors

Default value: DYNAMIC
Alias: CURSOR=

Syntax

CURSOR_TYPE=DYNAMIC | FORWARD_ONLY | KEYSET_DRIVEN | STATIC

Details Not all drivers support all cursor types. An error is returned if the specified cursor type is not supported.

By default, CURSOR_TYPE=DYNAMIC, but the driver is allowed to modify the default without an error.

If CURSOR_TYPE=DYNAMIC, then the cursor reflects all of the changes that are made to the rows in a result set as you scroll around the cursor. The data values and the membership of rows in the cursor can change dynamically on each fetch.

If CURSOR_TYPE=FORWARD_ONLY, then the cursor behaves like a DYNAMIC cursor except that it only supports fetching the rows sequentially.

If CURSOR_TYPE=KEYSET_DRIVEN, then the cursor determines which rows belong to the result set when the cursor is opened. However, changes that are made to these rows will be reflected as you scroll around the cursor.

If CURSOR_TYPE=STATIC, then the cursor builds the complete result set when the cursor is opened. No changes made to the rows in the result set after the cursor is opened will be reflected in the cursor. Static cursors are read only.

DBINDEX=

Indicates whether or not SAS calls the DBMS to find index(es) on the specified table.

Default value: NO
See Also: DBKEY=

Syntax

DBINDEX= YES | NO |<'>*index-name*<'>

Details

For a full description of this option, refer to Chapter 4, "SAS/ACCESS Data Set Options".

DBNULL=

Indicates whether or not NULL is a valid value for the specified variables or columns.

Default value: YES

Syntax

DBNULL= (<*column-name-1*=YES | NO > <...<*column-name-n*=YES | NO >>)

Details

For a full description of this option, refer to Chapter 4, "SAS/ACCESS Data Set Options".

DBSASTYPE=

Specifies data type(s) to override the default SAS data type(s) during input processing of data from DB2.

Default value: Varies by data type.

Syntax

DBSASTYPE=*(<column-name-1=<'>SAS-data-type<'>>*
<...<column-name-n=<'>SAS-data-type<'>>>)

column-name
specifies a DBMS column name.

SAS-data-type
specifies one of the following SAS data types:

- CHAR(n)
- NUMERIC
- DATETIME
- DATE
- TIME

Details This option is valid only when you read DB2 data into SAS.

By default, the SAS/ACCESS Interface to DB2 converts each DB2 data type to a predetermined SAS data type when processing data from DB2. When you need a

different data type, you can use DBSASTYPE= to override the default data type chosen by the SAS/ACCESS engine. SAS forces DB2 to perform the data conversions. Some conversions might not be supported; if a conversion is not supported, SAS prints an error to the log.

In the following example, DBSASTYPE= specifies a data type to use for the column MYCOLUMN when printing the DBMS data in SAS. If the data in this DBMS column is stored in a format that SAS does not support, such as DECIMAL(20), this enables SAS to print the values.

```
proc print data=mylib.mytable
   (DBSASTYPE=(mycolumn='CHAR(20)'));
run;
```

See "LIBNAME Statement Data Conversions" on page 29 for more details on the default data types for DB2.

DBTYPE=

Specifies data type(s) to override the default DB2 data type(s) when SAS outputs data to DB2.

Default value: VARCHAR(size) is the default for SAS character variables where size is derived from the length of the SAS variable. DATE is the default for SAS date variables, TIME is the default for SAS time variables, TIMESTAMP is the default for SAS datetime variables, and DOUBLE is the default for all other SAS numeric variables.

Syntax

DBTYPE= *(<column-name-1=<'>DBMS-type<'>>*
<...<column-name-n=<'>DBMS-type<'>>>)

Details

For a full description of this option, refer to Chapter 4, "SAS/ACCESS Data Set Options".

QUERY_TIMEOUT=

Specifies the number of seconds of inactivity to wait before canceling a query.

Default value: 0

Alias: TIMEOUT=

Syntax

QUERY_TIMEOUT= *number-of-seconds*

Details

The default value of 0 indicates that there is no time limit for a query. This option is useful when you are testing a query or if you suspect that a query might contain an endless loop.

READ_ISOLATION_LEVEL=

Default value: CS

Alias: RIL=

Syntax

READ_ISOLATION_LEVEL= RR | RS | CS | UR

RR = Repeatable Read
RS = Read Stability
CS = Cursor Stability
UR = Uncommitted Read

Details

The degree of isolation identifies

- the degree with which rows that are read and updated by the current application are available to other concurrently executing applications
- the degree with which update activity of other concurrently executing application processes can affect the current application.

The DB2 database manager supports four isolation levels. Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. Thus, all isolation levels ensure that any row that is changed by this application process during a unit of work is not changed by any other application process until the unit of work is complete. The isolation levels are defined in terms of several possible occurrences:

- Dirty read — A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it will be able to see changes that are made by those concurrent transactions even before they commit.

  For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.
- Nonrepeatable read — If a transaction exhibits this phenomenon, it is possible that it may read a row once and, if it attempts to read that row again later in the course of the same transaction, the row might have been changed or even deleted by another concurrent transaction. Therefore, the read is not (necessarily) repeatable.

  For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

- Phantom reads — When a transaction exhibits this phenomemon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

  For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it will see a row that did not previously exist, a phantom.

The isolation levels for READ_ISOLATION_LEVEL= include the following:

- Repeatable Read (RR)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - does not allow phantom reads
- Read Stability (RS)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - allows phantom reads
- Cursor Stability (CS)
  - does not allow dirty reads
  - allows nonrepeatable reads
  - allows phantom reads

  This is the default value for DB2.
- Uncommitted Read (UR)
  - allows dirty reads
  - allows nonrepeatable reads
  - allows phantom reads

READ_ISOLATION_LEVEL= is ignored if READ_LOCK_TYPE= is not set to ROW.

READ_LOCK_TYPE=

Specifies how a table is locked during a READ operation.

Default value: ROW

Syntax

READ_LOCK_TYPE= ROW | TABLE

Details

If you specify ROW, the locking is determined by the READ_ISOLATION_LEVEL= setting. If you specify TABLE, the entire table is locked in SHARE mode.

For a full description of this option, refer to Chapter 4, "SAS/ACCESS Data Set Options".

ROWSET_SIZE=

Specifies the number of rows to use when reading data from the DBMS.

Default value: 0

Syntax

ROWSET_SIZE= *number-of-rows*

Details

By default, ROWSET_SIZE=0, so that no internal SAS buffering is performed. Setting ROWSET_SIZE=0 causes the SQLFetch API call to be used.

When ROWSET_SIZE=1, only one row is retrieved at a time. The higher the value for ROWSET_SIZE=, the more rows the DB2 engine retrieves in one fetch operation. This option reduces the amount of I/O that is used and can help improve performance. However, because SAS software stores the rows in memory, higher values for ROWSET_SIZE= use more memory. In addition, if too many rows are selected at once, then the rows that are returned to the SAS application might be out of date. For example, if someone else modified the rows, you would not see the changes. Setting ROWSET_SIZE=1 or greater causes the SQLExtendedFetch API call to be used.

SASDATEFMT=

Changes the SAS date format of a DBMS column.

Default value: None

Syntax

SASDATEFMT= *(DBMS-date-col='SAS-date-format' ...)*

Details

For a full description of this option, refer to Chapter 4, "SAS/ACCESS Data Set Options".

SCHEMA=

Enables you to read database objects, such as tables and views, in the specified schema.

Default value: None

Syntax

SCHEMA= *schema-name*

Details

A schema is a logical classification of objects in a database. SCHEMA= is optional. If it is omitted, you connect to the default schema. In the following example, the SCHEMA= option causes MYDB.TEMP_EMPS to be interpreted by DB2 as SCOTT.TEMP_EMPS.

```
proc print data=mydb.temp_emps
    SCHEMA=scott;
run;
```

UPDATE_ISOLATION_LEVEL=

Defines the degree of isolation of the current application process from other concurrently running application processes.

Default value: CS

Alias: UIL=

Syntax

UPDATE_ISOLATION_LEVEL= CS | RS | RR

CS = Cursor Stability
RS = Read Stability
RR = Repeatable Read

Details

The degree of isolation identifies

- the degree with which rows that are read and updated by the current application are available to other concurrently executing applications
- the degree with which update activity of other concurrently executing application processes can affect the current application.

The DB2 database manager supports three isolation levels. Regardless of the isolation level, the database manager places exclusive locks on every row that is inserted, updated, or deleted. Thus, all isolation levels ensure that any row that is changed by this application process during a unit of work is not changed by any other application process until the unit of work is complete. The isolation levels are defined in terms of several possible occurrences:

- Dirty read — A transaction that exhibits this phenomenon has very minimal isolation from concurrent transactions. In fact, it will be able to see changes that are made by those concurrent transactions even before they commit.

  For example, suppose that transaction T1 performs an update on a row, transaction T2 then retrieves that row, and transaction T1 then terminates with rollback. Transaction T2 has then seen a row that no longer exists.

- Nonrepeatable read — If a transaction exhibits this phenomenon, it is possible that it may read a row once and, if it attempts to read that row again later in the course of the same transaction, the row might have been changed or even deleted by another concurrent transaction. Therefore, the read is not (necessarily) repeatable.

  For example, suppose that transaction T1 retrieves a row, transaction T2 then updates that row, and transaction T1 then retrieves the same row again. Transaction T1 has now retrieved the same row twice but has seen two different values for it.

- Phantom reads — When a transaction exhibits this phenomemon, a set of rows that it reads once might be a different set of rows if the transaction attempts to read them again.

  For example, suppose that transaction T1 retrieves the set of all rows that satisfy some condition. Suppose that transaction T2 then inserts a new row that satisfies that same condition. If transaction T1 now repeats its retrieval request, it will see a row that did not previously exist, a phantom.

The isolation levels for UPDATE_ISOLATION_LEVEL= include the following:

- Repeatable Read (RR)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - does not allow phantom reads
- Read Stability (RS)
  - does not allow dirty reads
  - does not allow nonrepeatable reads
  - allows phantom reads
- Cursor Stability (CS)
  - does not allow dirty reads
  - allows nonrepeatable reads
  - allows phantom reads

This is the default value for DB2.

UPDATE_ISOLATION_LEVEL= is ignored if UPDATE_LOCK_TYPE= is not set to ROW.

UPDATE_LOCK_TYPE=

Specifies how a DB2 table is locked during an UPDATE operation.

Default value: ROW

Syntax

UPDATE_LOCK_TYPE= ROW | TABLE

Details

If you specify ROW, the locking is determined by the UPDATE_ISOLATION_LEVEL= setting. If you specify TABLE, the entire table is locked in exclusive mode.

For a full description of this option, refer to Chapter 4, "SAS/ACCESS Data Set Options".

See Also

READ_LOCK_TYPE=

ACCESS Procedure: DB2 Specifics

This section describes the statements that you use in the SAS/ACCESS Interface to DB2 under the OS/2 operating environment.

Operating Environment Information: PROC ACCESS is valid only for DB2 running under OS/2. It is not valid under any other operating environment. △

ACCESS Procedure Statements for DB2

To create an access descriptor, you use database identification statements that supply DBMS-specific information to the SAS System. These database identification statements must immediately follow the CREATE statement that specifies the access descriptor to be created.

Database identification statements are required only when you create access descriptors. Because DB2 information is stored in an access descriptor, you do not need to repeat this information when you create view descriptors.

The SAS/ACCESS Interface to DB2 uses the following procedure statements in batch mode:

PROC ACCESS DBMS=DB2 <*view-descriptor-options*>;
CREATE <*libref.*>*member-name.* ACCESS|VIEW;
UPDATE <*libref.*>*member-name.* ACCESS|VIEW;
IN|DATABASE|DSN=<'>*database-name*<'>;
TABLE=<'>*schema-name.table-name*<'>
ASSIGN | AN<=>YES|NO;
DROP<'>*column-identifier-1*<'><…<'>*column-identifier-n*<'>>;
FORMAT | FMT<'>*column-identifier-1*<'><=>*SAS-format-name-1*
<…<'>*column-identifier-n*<'><=>*SAS-format-name-n*>;
LIST<ALL|VIEW|<'>*column-identifier-1*><'>;
QUIT | EXIT;
RENAME<'>*column-identifier-1*<'><=>*SAS-variable-name-1*
<…<'>*column-identifier-n*><'><=>*SAS-variable-name-n*;

RESETALL | <'>*column-identifier-1*<'><...<'>*column-identifier-n*<'>>;

SELECT ALL | <'>*column-identifier-1*<'><...<'>*column-identifier-n*<'>>;

SUBSET*selection-criteria*;

UNIQUE | UN<=>YES | NO;

RUN;

IN | DATABASE | DSN= <'>*database-name*<'>;
specifies the name of the database where the DB2 table resides. *Database name* is limited to eight characters. The IN statement is required and follows the CREATE statement. DATABASE= and DSN= are aliases for the IN statement.

The database that you specify must already exist. If the database name contains the following special characters (_,$,@,#), you must enclose it in quotes. However, DB2 recommends against using special characters in database names.

TABLE= <'><*schema-name.*>*table-name*<'>;
identifies the DB2 table or DB2 view that you want to use to create an access descriptor. *Table name* is limited to 18 characters. If you quote the name, it is case-sensitive. The TABLE= statement is required.

Schema-name is a person's name or group ID that is associated with the DB2 table. The schema name is limited to eight characters.

The following example creates an access descriptor and a view descriptor that are based on DB2 data. For the DBMS= option in the PROC ACCESS statement, use **db2**.

```
options linesize=80;
/* create access descriptor  */

proc access dbms=db2;
   create adlib.customr.access;
   in sample; user=testuser; password=testpass;
   table=sasdemo.customers;
   assign=yes;
   rename customer=custnum;
   format firstorder date9.;
   list all;

   /* create usacust view     */
   create vlib.usacust.view;
   select customer state zipcode name
          firstorder;
   subset where customer like '1%';
run;
```

See Chapter 9, "ACCESS Procedure Reference" for more information.

DBLOAD Procedure: DB2 Specifics

The DBLOAD procedure enables you to create and load a DBMS table from a SAS data set. This section describes the statements that you use in the SAS/ACCESS Interface to DB2. PROC DBLOAD is valid for DB2 running under HP-UX, SUN, R6000, OS/2, Windows95, and WinNT operating environments.

DBLOAD Procedure Statements for DB2

To create and load a DB2 table, the SAS/ACCESS Interface to DB2 uses the following statements in batch mode.

PROC DBLOAD DBMS=DB2 <DATA=<*libref.*>*SAS-data-set*><APPEND>;

IN|DATABASE|DSN=<'>*database-name*<'>;

USER|UID=<'>*username*<'>;

PASSWORD|PASS|PWD|PW|
USING=<'>*password*<'>;

TABLE=<'><*schema-name.*>*table-name*<'>;

ACCDESC | ACCESS | AD=<*libref.*>*access-descriptor*;

COMMIT=*commit-frequency*;

DELETE*variable-identifier-1*<*...variable*
-identifier-n;>

ERRLIMIT=*error-limit*;

LABEL;

LIMIT=*load-limit*;

LIST<ALL|COLUMN|*variable-identifier*>;

NULLS*variable-identifier-1* = Y|N|D<*...variable-identifier-n= Y|N*>;

QUIT | EXIT;

RENAME | COLUMN*variable-identifier-1*=<'>*column-name-1*<'>
<*...variable-identifier-n* = <'>*column-name-n* <'>>;

RESET ALL | *variable-identifier-1*<*...variable-identifier-n*>;

SQL *DBMS-specific SQL-statement*;

TYPE *variable-identifier-1='column-type-1'*
<*...variable-identifier-n = 'column-type-n'*>;

WHERE*SAS-where-expression*;

LOAD;

RUN;

IN | DATABASE | DSN= <'>*database-name*<'>;
specifies the name of the database in which you want to store the new DB2 table. The IN statement is required and must immediately follow the PROC DBLOAD statement. *Database name* is limited to eight characters. DATABASE= is an alias for the IN statement.

The database that you specify must already exist. If the database name contains the following special characters (_,$,@,#), you must enclose it in quotes. However, DB2 recommends against using special characters in database names.

USER | UID= <'>*username* <'>;
enables you to connect to a DB2 database, such as SQL Server or AS/400, with a user ID that is different from the default login ID.

The USER= and PASSWORD= statements are optional in DB2. If you specify USER=, you must also specify PASSWORD=. If USER= is omitted, your default user ID is used.

Operating Environment Information: The USER= statement does not apply if you are running DB2 under OS/2. △

PASSWORD | PASS | PWD | PW | USING= <'>*password*<'>;
specifies the DB2 password that is associated with your user ID.

The USER= and PASSWORD= statements are optional in DB2 because users have default user IDs. If you specify USER=, you must specify PASSWORD=.

Operating Environment Information: The PASSWORD= statement does not apply if you are running DB2 under OS/2. △

TABLE=<'><*schema-name.*>*table-name*<'>;
identifies the DB2 table or DB2 view that you want to use to create an access descriptor. *Table name* is limited to 18 characters. If you quote the name, it is case-sensitive. A DB2 table with the same name cannot already exist. The TABLE= statement is required.

The *schema-name* is a person's name or group ID that is associated with the DB2 table. The schema name is limited to eight characters and is required in batch mode.

ACCDESC | ACCESS | AD=<*libref.*>*access-descriptor*;
creates an access descriptor that is based on the DB2 table that you are creating and loading.

Operating Environment Information: The ACCDESC= statement applies only if you are running DB2 under OS/2. △

See Chapter 10, "DBLOAD Procedure Reference" for more information.

DBLOAD Procedure Examples

The following example creates a new DB2 table, SASDEMO.EXCHANGE, from the MYDBLIB.RATEOFEX data file on an OS/2 platform. An access descriptor ADLIB.EXCHANGE is also created, based on the new table. You must be granted the appropriate privileges in order to create new DB2 tables or views. For the DBMS= option of the PROC DBLOAD procedure, use **db2**.

```
proc dbload dbms=db2 data=mydblib.rateofex;
   in=sample; user=testuser; password=testpass;
   table=sasdemo.exchange;
   accdesc=adlib.exchange; /* only applies to OS/2 */
   rename fgnindol=fgnindollars
          4=dollarsinfgn;
   nulls updated=n fgnindollars=n
         dollarsinfgn=n country=n;
   load;
run;
```

The next example sends only a DB2 SQL GRANT statement to the SAMPLE database and does not create a new table. Therefore, the TABLE= and LOAD statements are omitted.

```
proc dbload dbms=db2;
   in=sample;
   sql grant select on sasdemo.exchange
      to testuser;
run;
```

SQL Procedure Pass-Through Facility: DB2 Specifics

The SQL Procedure Pass-Through Facility enables you to connect to and disconnect from a DBMS, to send DBMS specific statements to the DBMS, and to retrieve DBMS data for your SAS programs. This section describes the DBMS-specific arguments that you use in the CONNECT statement"CONNECT Statement "on page 23. For a complete description of the SQL Procedure Pass-Through Facility, see Chapter 6, "SQL Procedure's Interaction with SAS/ACCESS Software" .

CONNECT Statement

Establishes a connection with the DBMS

Syntax

CONNECT TO DB2 <AS *alias*> <(*DB2-connection-arguments*)>;

Arguments

Use the following arguments with the CONNECT statement:

alias
specifies an optional alias that has 1 to 32 characters. If you specify an alias, the keyword AS must appear before the alias.

(*DB2-connection-arguments*)
specifies the DBMS-specific arguments that are needed by PROC SQL in order to connect to the DBMS. These arguments must be enclosed in parentheses. For some databases, these arguments have default values and therefore are optional. The arguments for DB2 are described in the following sections.

DB2 Connection Arguments You must specify the PROC SQL CONNECT statement when you connect to DB2. You can connect to only one DB2 database at a time; however, you can use multiple CONNECT statements to connect to multiple DB2 data sources by using the *alias* argument to distinguish your connections.

The following list describes the arguments that are used for DB2 in the CONNECT statement. You can use the arguments DATABASE=, USER=, and PASSWORD= to connect to most data sources. Use the PROMPT=, NOPROMPT=, COMPLETE=, REQUIRED=, or AUTOCOMMIT= arguments to provide additional information or to select and connect to the data source.

There are multiple ways that you can connect to the DBMS when using the CONNECT statement. Use only one of the following methods for each connection since they are mutually exclusive:

- specify USER=, PASSWORD=, and DATABASE=, or
- specify COMPLETE=, or
- specify NOPROMPT=, or
- specify PROMPT=, or
- specify REQUIRED=.

DATABASE | DATASRC | DSN | DS =<'>*database-name*<'>
specifies the name of the database that you want to connect to; the name is limited to eight characters. Specify either DSN= or one (and only one) of the following arguments: PROMPT=, NOPROMPT=, COMPLETE=, or REQUIRED= . These arguments are all mutually exclusive of each other.

USER | UID=<'>*username*<'>
specifies the DBMS password.

The USER= and PASSWORD= connections are optional in DB2. If you specify USER=, you must also specify PASSWORD=.

PASSWORD | PW | PASS | PWD | USING=<'>*password*<'>
specifies the DB2 password that is associated with your user ID.

The USER= and PASSWORD= connection options are optional in DB2 because users may have default user IDs. If you specify USER=, you must specify PASSWORD=.

AUTOCOMMIT=YES | NO
indicates whether or not updates are committed immediately after they are submitted.

If AUTOCOMMIT=YES, no rollback is possible. This is the default for the SQL Procedure Pass-Through Facility and read-only connections.

If AUTOCOMMIT=NO, the SAS/ACCESS engine automatically does the commit when it reaches the end of the file.

COMPLETE=<'>*CLI-connection-string*<'>
specifies connection options for your data source or database. If you specify enough correct connection options, the SAS/ACCESS engine connects to your data source or database. Otherwise, you are prompted for the connection options with a dialog box. Separate multiple options with a semicolon. When a successful connection is made, the complete connect string is returned in the SYSDBMSG and SQLXMSG macro variables.

COMPLETE= is similar to the PROMPT= option. However, if COMPLETE= attempts to connect and fails, then a dialog box is displayed and you can edit values or enter additional values.

COMPLETE= is optional.

See your driver documentation for more details.

NOPROMPT=<'>*CLI-connection-string* <'>
specifies connection options for your data source or database. Separate multiple options with a semicolon. If you specify enough correct connection options, the SAS/ACCESS engine connects to the data source or database. Otherwise, an error is returned and no dialog box is displayed. NOPROMPT= is optional. If it is omitted and other options settings are not specified, the default settings are used.

PROMPT=<'> *CLI-connection-string*<'>
specifies connection options to the data source.

A dialog box is displayed, using the values from the PROMPT= connection string. You can edit any field before you connect to the data source. When a successful connection is made, the complete connect string is returned in the SYSDBMSG and SQLXMSG macro variables.

PROMPT= is similar to the COMPLETE= option. However, unlike COMPLETE=, PROMPT= does not attempt to connect to the DBMS first. It displays the dialog box where you can edit or enter additional values.

REQUIRED=<'>*CLI-connection-string*<'>
specifies connection options for your data source or database. Separate multiple options with a semicolon.

If you specify enough correct connection options, such as user ID, password, and data source name, the SAS/ACCESS engine connects to the data source or database. Otherwise, a dialog box is displayed to prompt you for the connection options. Options in the dialog box that are not related to the connection are disabled. REQUIRED= only allows you to modify required fields in the dialog box. When a successful connection is made, the complete connect string is returned in the SYSDBMSG and SQLXMSG macro variables.

REQUIRED= is similar to COMPLETE= because it attempts to connect to the DBMS first. However, if REQUIRED= attempts to connect and fails, then a dialog box is displayed and you can only edit values in the required fields.

REQUIRED= is optional.

CONNECT Examples The following example connects to the SAMPLE database and sends it two EXECUTE statements to process.

```
proc sql;
   connect to db2 (database=sample);
   execute (create view
               sasdemo.whotookorders as
            select ordernum, takenby,
                   firstname, lastname, phone
              from sasdemo.orders,
                   sasdemo.employees
              where sasdemo.orders.takenby=
                    sasdemo.employees.empid)
           by db2;
   execute (grant select on
            sasdemo.whotookorders to testuser)
           by db2;
   disconnect from db2;
quit;
```

The next example connects to the SAMPLE database by using an alias (DB1) and performs a query, shown in italic type, on the SASDEMO.CUSTOMERS table.

```
proc sql;
   connect to db2 as db1 (database=sample);
   select *
      from connection to db1
         (select * from sasdemo.customers
             where customer like '1%');
   disconnect from db1;
quit;
```

DB2 Naming Conventions

DB2 objects include tables, views, columns, and indexes. Use the following naming conventions for them:

- A name can start with a letter or one of the following symbols: the dollar sign ($), the number (or pound) sign (#), or the at symbol (@).
- A name can be from 1 to 18 characters long.

- A name can contain the letters A through Z, any valid letter with an accent (such as a), the digits 0 through 9, the underscore (_), the dollar sign ($), the number or pound sign (#), or the at symbol (@).
- A name is not case-sensitive (for example, the table name CUSTOMERS is the same as Customers), but object names are converted to uppercase when typed. If a name is enclosed in quotes, then the name is case-sensitive.
- A name cannot be a DB2 or an SQL reserved word, such as WHERE or VIEW.
- A name cannot be the same as another DB2 object that has the same type.

Schema and database names have similar conventions, except that they are each limited to eight characters. For more information, see your DB2 SQL reference manual.

DB2 Data Types

Every column in a table has a name and a data type. The *data type* tells DB2 how much physical storage to set aside for the column and the form in which the data are stored. DB2 uses IBM SQL data types. The data types fall into three categories: types for string data, types for numeric data, and types for datetime values. Each of these types is described in the following sections. For more information about DB2 data types, see your DB2 SQL reference manual.

Note: The SAS/ACCESS interface does not support the following DB2 data types: BLOB, CLOB, and DBCLOB. △

String Data

CHAR(*n*)
specifies a fixed-length column for character string data. The maximum length is 254 characters.

VARCHAR(*n*)
specifies a varying-length column for character string data. The maximum length of the string is 4000 characters. If the length is greater than 254, the column is a long-string column. SQL imposes some restrictions on referencing long-string columns. For more information about these restrictions, see your IBM documentation.

LONG VARCHAR
specifies a varying-length column for character string data. The maximum length of a column of this type is 32700 characters. A LONG VARCHAR column cannot be used in certain functions, subselects, search conditions, and so forth. For more information about these restrictions, see your IBM documentation.

GRAPHIC(*n*)
specifies a fixed-length column for graphic string data. *n* specifies the number of double-byte characters and can range from 1 to 127. If *n* is not specified, the default length is 1.

VARGRAPHIC(*n*)
specifies a varying-length column for graphic string data. *n* specifies the number of double-byte characters and can range from 1 to 2000.

LONG VARGRAPHIC
specifies a varying-length column for graphic-string data. *n* specifies the number of double-byte characters and can range from 1 to 16350.

Numeric Data

SMALLINT
specifies a small integer. Values in a column of this type can range from -32768 through +32767.

INTEGER
specifies a large integer. Values in a column of this type can range from -2147483648 through +2147483647.

FLOAT | DOUBLE | DOUBLE PRECISION
specifies a floating-point number that is 64 bits long. Values in a column of this type can range from –1.79769E+308 to –2.225E–307 or +2.225E–307 to +1.79769E+308, or they can be 0. (This data type is stored the same way that the SAS System stores its numeric data type; therefore, numeric columns of this type require the least processing when they are being accessed by the SAS System.)

DECIMAL | DEC | NUMERIC | NUM
specifies a mainframe packed decimal number with an implicit decimal point. The position of the decimal point is determined by the precision and scale of the number. The scale, which is the numbers to the right of the decimal point, cannot be negative or greater than the precision. The maximum precision is 31 digits. Note that numbers that require decimal precision greater than 15 digits may be subject to rounding and conversion errors.

Dates, Times, and Timestamps

SQL date and time data types are collectively called datetime values. The SQL data types for dates, times, and timestamps are listed here. Be aware that columns of these data types may contain data values that are out of range for the SAS System.

DATE
specifies date values in various formats, as determined by the country code of the database. For example, the default format for the United States is *mm-dd-yyyy* and the European standard format is *dd.mmm.yyyy*. The range is 01-01-0001 to 12-31-9999. A date always begins with a digit, is at least eight characters long, and is represented as a character string. For example, in the U.S. default format, January 25, 1991, would be input as 01-25-1991.

The entry format can vary according to the edit codes that are associated with the field. For more information about edit codes, see your IBM documentation.

TIME
specifies time values in a three part format. The values range from 0 to 24 for hours (*hh*) and from 0 to 59 for minutes (*mm*) and seconds (*ss*). The default form for the United States is *hh:mm:ss*, and the IBM European standard format for time is *hh.mm[.ss]*. For example, in the U.S. default format 2:25 p.m. would be input as 14:25:00.

The entry format can vary according to the edit codes that are associated with the field. For more information about edit codes, see your IBM documentation.

TIMESTAMP
combines a date and time and adds an optional microsecond to make a seven part value of the format *yyyy-mm-dd-hh.mm.ss[.nnnnnn]*. For example, a timestamp for precisely 2:25 p.m. on January 25, 1991, would be 1991-01-25-14.25.00.000000. Values in a column of this type have the same ranges as described earlier for DATE and TIME.

For more information about SQL data types, datetime formats, and edit codes that are used in the United States and other countries, see your IBM documentation.

Null Values

DB2 has a special value called NULL. NULL means that a value in a row is not known or is missing; it does not mean the value is blank or zero. It is analogous to the SAS System's missing value.

You can define a column in a table so that it requires data. To do this in SQL, you specify a column as NOT NULL. NOT NULL tells SQL to only allow a row to be added to a table if there is a value for the field. For example, NOT NULL assigned to the field CUSTOMER in the table SASDEMO.CUSTOMER does not allow a row to be added unless there is a value for CUSTOMER.

Columns can also be defined as NOT NULL WITH DEFAULT. For more information about using the NOT NULL WITH DEFAULT value, see your DB2 SQL reference manual.

Knowing whether a DB2 column allows NULLs, or whether the host system supplies a default value for a column that is defined as NOT NULL WITH DEFAULT, can assist you in writing selection criteria and in entering values to update a table. Unless a column is defined as NOT NULL or NOT NULL WITH DEFAULT, it allows NULL values.

For more information, see “DBNULL=” on page 12 and Chapter 4, "SAS/ACCESS Data Set Options".

ACCESS Procedure Data Conversions

The following table shows the default SAS System variable formats that the ACCESS procedure assigns to each DB2 data type.

Operating Environment Information: PROC ACCESS is valid only for DB2 running under OS/2. It is not valid under any other operating environment. △

Table 1.1 Default SAS System Variable Formats for DB2 Data Types

DB2 for Common Servers Data Type	SAS Variable Format
Character (fixed length)	$*w*.(*n*<32,767)*
	$32767. (*n*>32,767)
Character (varying length)	$*w*.(*n*<32,767)*
	$32767. (*n*>32,767)
INTEGER	11.0
SMALLINT	6.0

DB2 for Common Servers Data Type	SAS Variable Format
DECIMAL	*p*+2.*s*. for example, DEC(6,4)=8.4
FLOAT	none
TIME	TIME11.2.
DATE	DATE9.
TIMESTAMP	DATETIME25.6

* *n* in DB2 data types is equivalent to *w* in SAS formats.

If DB2 data fall outside of the valid SAS data ranges, you get an error message in the SAS log when you try to read the data.

DBLOAD Procedure Data Conversions

The following table shows the default DB2 data types that the DBLOAD procedure assigns to SAS variable formats.

Table 1.2 PROC DBLOAD: Default DB2 Data Types for SAS System Variable Formats

SAS Variable Format	DB2 for Common Servers Data Type
$*w*.	CHAR(*n*)
w.	DECIMAL(*p*)
w.d	DECIMAL(*p*,*s*)
IB*w.d*, PIB*w.d*	INTEGER
all other numerics*	DOUBLE
datetime*w.d*	TIMESTAMP
date*w*.	DATE
time.**	TIME

* Includes all SAS numeric formats, such as BINARY8 and E10.0.
** Includes all SAS time formats, such as TOD*w.d* and HHMM*w.d*.

LIBNAME Statement Data Conversions

The following table shows the default SAS System variable formats that the LIBNAME statement assigns to DB2 data types during input operations.

Table 1.3 LIBNAME Statement: Default SAS Formats for DB2 Data Types

DB2 for Common Servers Data Type	SAS Data Type	Default SAS Format
CHAR(*n*)	character	$*n*.
VARCHAR(*n*)	character	$*n*.

DB2 for Common Servers Data Type	SAS Data Type	Default SAS Format
LONG VARCHAR	character	$*n*.
GRAPHIC(*n*), VARGRAPHIC(*n*), LONG VARGRAPHIC	character	$*n*.
INTEGER	numeric	11.
SMALLINT	numeric	6.
DECIMAL	numeric	*m*.*n*
NUMERIC	numeric	*m*.*n*
FLOAT	numeric	none
DOUBLE	numeric	none
TIME	numeric	TIME8.
DATE	numeric	DATE9.
TIMESTAMP	numeric	DATETIME*m*.*n*

* *n* in DB2 data types is equivalent to *w* in SAS formats.

The following table shows the default DB2 data types that the LIBNAME statement assigns to SAS variable formats during output operations.

Table 1.4 LIBNAME Statement: Default DB2 Data Types for SAS Variable Formats

SAS Variable Format	DB2 for Common Servers Data Type
m.*n*	DECIMAL (*m*,*n*)
other numerics	DOUBLE
$*n*.	VARCHAR(*n*) (*n*<=4000) LONG VARCHAR(*n*) (*n*>4000)
datetime formats	TIMESTAMP
date formats	DATE
time formats	TIME

* *n* in DB2 data types is equivalent to *w* in SAS formats.

Your Turn

If you have comments or suggestions about *SAS/ACCESS® Software for Relational Databases: Reference, Version 8 (DB2® under UNIX and PC Hosts Chapter)*, please send them to us on a photocopy of this page or send us electronic mail.

For comments about this book, please return the photocopy to

SAS Institute
Publications Division
SAS Campus Drive
Cary, NC 27513
email: yourturn@sas.com

For suggestions about the software, please return the photocopy to

SAS Institute
Technical Support Division
SAS Campus Drive
Cary, NC 27513
email: suggest@sas.com

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ACCESS® Software for Relational Databases: Reference, Version 8 (DB2® under UNIX and PC Hosts Chapter)*, Cary, NC: SAS Institute Inc., 1999.

SAS/ACCESS® Software for Relational Databases: Reference, Version 8 (DB2® under UNIX and PC Hosts Chapter)

ISBN 1–58025–538–8

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, October 1999